REPORT DOCUMENTAT

AD-A286 848

Approved
No. 0704-0188

Public reporting burden for this collection of information is estimated to aver gathering and maintaining the data needed, and completing and reviewing th collection of information, including suggestions for reducing this burden, to V Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Ma

searching existing data sources, ate or any other aspect of this ns and Reports, 1215 Jefferson gton, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	27 Oct 95	

4. TITLE AND SUBTITLE
Tools for Satellite Ground Track and Coverage Analysis

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael Bottner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Students Attending:
University of Colorado

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA-95-122

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STREET, BLDG 125
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW AFR 190-1
Distribution Unlimited
BRIAN D. Gauthier, MSgt, USAF
Chief Administration

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

95-02341

95 11 a 052

DTIC QUALITY INSPECTED 8

14. SUBJECT TERMS	15. NUMBER OF PAGES 34
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)

Tools for Satellite

Ground Track and Coverage Analysis

Prepared by

Michael Bettner

MS Engineering Space Operations Option

October 1, 1995

ABSTRACT

More and more we are relying on satellites. With the advancements in technology, we will soon be using satellites for things never dreamed of before. In today's world, some of these advancements will be tied to saving money in other areas. This forms the foundation of many projects looking into offboard sensor technology. The goal is to save money by using space assets to replace or supplement sensors on aircraft, ships, or any other system.

The author has worked on several projects relating to offboard sensors. The software we are presenting here has been a valuable tool. This software package is written for Microsoft Excel. It enable the user to enter orbital information and viewing angles of a satellite. The program will then provide several plots. These plots include the satellite ground track, the coverage region of the satellite over a time interval, and the instantaneous field of view for any time. Additionally the user can enter a square target region of any size with its center at a particular latitude and longitude.

With this software package the user will be able to look at various orbits and viewing characteristics and easily see the impact of the choices on the coverage of different geographic regions. This is very helpful for the initial stages of designing an orbit for a satellite system. It is also helpful in analyzing the coverage of existing satellite's for different regions. Finally, this software package can be useful for students just entering the study of Astrodynamics.

Table of Contents

1. BACKGROUND 1

2. INTRODUCTION 2

3. ASTRODYNAMICS 3

3.1 Equations of Motion 3
3.2 Orbital Elements to Position Vector 6
3.3 Orbit Propagation 7
3.3.1 Kepler Problem 7
3.3.2 Perturbations 10
3.4 Position Vector to Geodetic Latitude and Longitude 12
3.5 Satellite Coverage 18
3.5.1 Swath Ground Track 18
3.5.2 Instantaneous Field of View 22

4. SOFTWARE 23

4.1 Excel Software 23
4.2 Software Setup 24
4.3 Input Data 25
4.4 Sample Plots 27
4.5 Limitations 31
4.6 Applications 32

5. CONCLUSION 33

ENDNOTES

APPENDIX A: COORDINATE FRAME TRANSFORMATION PQW TO IJK

APPENDIX B: GEODETIC LATITUDE SOLUTIONS
QUARTIC SOLUTION
ITERATIVE SOFTWARE SOLUTION
COMPARISON OF ANALYTIC WITH ITERATIVE SOLUTION
MAPLE V SOLUTION FOR SPECIFIC EXAMPLE

APPENDIX C: VISUAL BASIC CODE
MODULE 1: CALCULATION CODE FOR PLOT.XLS
MODULE 2: DIALOG BOX CODE FOR PLOT.XLS

Table of Figures

FIGURE 1 ORBITAL ELEMENTS AND **IJK** FRAME 5

FIGURE 2 RELATIONSHIP BETWEEN **PQW** AND **IJK** 6

FIGURE 3 GEODETIC VERSUS GEOCENTRIC LATITUDE 12

FIGURE 4 LOCAL SIDEREAL TIME AND XZ PLANE 16

FIGURE 5 GST, LST, AND EAST LONGITUDE RELATIONSHIP 17

FIGURE 6 GROUND TRACK AND SWATH COVERAGE 19

FIGURE 7 SATELLITE VIEWING ANGLES 20

FIGURE 8 SWATH VECTOR 21

FIGURE 9 INPUT INFORMATION SAMPLE 26

FIGURE 10 GROUND TRACK & INSTANTANEOUS COVERAGE PLOT 27

FIGURE 11 SWATH PLOT 28

FIGURE 12 IFOV VS TARGET REGION FOR SPECIFIC TIME 29

FIGURE 13 ORBIT RELATIVE TO EARTH 30

FIGURE 14 MOLNIYA PLOT USING SLOWPLOT.XLS 31

1. Background

With the advancement of space technology, we are performing many functions more efficiently from space. The potential to save money by accomplishing tasks from space is leading us to reconsider the way we conduct business. For example, we may soon find GPS replacing the thousands of radio navigation aids (TACAN, VOR, NDB, etc.) as the primary means of navigation for aircraft. With the current emphasis on cutting or limiting government spending, new projects may need to look to space for cost effective solutions.

One such area that is getting a lot of attention these days is the idea of offboard sensors. Think of the number of aircraft and ships worldwide which have the same types of sensors. We have radar, electronic sensors, and photographic capabilities. Yet we are accomplishing all of these functions from space as well. Thus we should examine the feasibility of using space assets to gather the information and relay this to different users.

If we can get the same quality of information from space to the user in a timely manner, there is the potential to save money. Take radar as an example. Most aircraft have a radar. It is heavy, requires a lot of maintenance, and is not cheap. If we are able to use space assets to provide some or all of the radar information to the aircraft, we may be able to use a smaller and cheaper radar on the aircraft, thus saving money.

The following pages contain the development of a software package which may be helpful in the initial stages of examining an offboard sensor problem. This package was developed for a project relating to the concept of offboard sensors.

2. Introduction

While working on a project relating to offboard sensors, the need for software to help with the design or evaluation became necessary. The project was to design a prototype system which could provide timely, tactical information that is key to fighting and winning isolated, short duration battles anywhere in the world below 70 degrees latitude. Basically we were looking for a satellite which could gather and relay information to a mobile user in a tactical theater environment. The idea was to gather or relay photo reconnaissance, electronic intelligence, weather, news, and force location to friendly forces. The requirement for the satellite to support a specific theater (launch on demand), perform photo reconnaissance, and act as a communications relay made the orbit design very difficult.

This software provides the means to analyze different satellite orbits and viewing angles versus a given target region. Thus if we have a battle or target region, we can visually compare the satellite ground track to the target region. Additionally, we can use different elevation angles (representing how much of the earth the satellite can "see" at a particular time) in order to examine when different satellite functions can be performed relative to the target region.

For example, lets assume our satellite can only take pictures up to a 45 degree elevation angle, but communications are possible up to a 5 degree elevation angle. The software will allow us to see which passes allow communications, which allow both photos and communications, and which passes will have no coverage. Additionally, we can step through minute by minute to get a snapshot or instantaneous coverage view of the satellite versus the

target region. This software is very useful in examining orbits and elevation angles and the impact of changes on the coverage of a given target region.

3. Astrodynamics

Sections 3.1 through 3.3 are a review of the basics of Astrodynamics. For the reader familiar with the two-body equations of motion, classical orbital elements, the Kepler problem, and the effect of perturbations on the orbital elements, you can may skip to section 3.4.

3.1 Equations of Motion

Before we can write software to generate satellite orbits, ground tracks, and coverage, we must mathematically describe a general satellite orbit. The following equations from *Fundamentals of Astrodynamics* form the starting point for describing 2-body motion (that of a satellite about a large body such as the Earth):[1]

$$\ddot{\bar{r}} + (\mu r^{-3})\,\bar{r} = 0 \tag{1}$$

$$r = p/(1 + e \cos \nu) \tag{2}$$

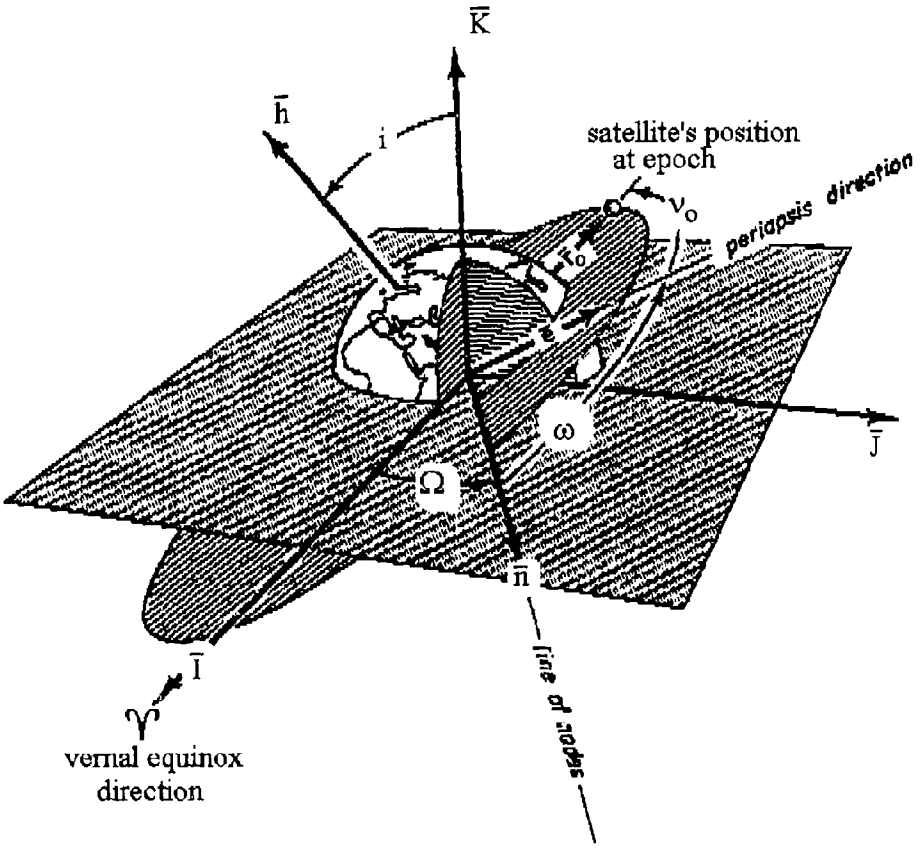
$$p = a(1 - e^2) \tag{3}$$

Equation (1) is the 2-body differential equation which describes the motion of a satellite around a large body. This equation assumes the only force acting on the satellite is gravity, the mass of the central body (the Earth in our case) is much greater than the mass of the satellite, the central body is a uniform sphere (that is it can be treated as a point mass), and there are only two bodies in the system (the Earth and the satellite) Equation (2) is a solution to equation (1) known as the

polar equation of a conic.[2] The parameter μ is known as the gravitational parameter and is the product of the universal gravitation constant, G, and the mass of the major attracting body, the earth in our case.[3]

All orbits (with the former assumptions) follow the path of a conic section (ellipse, parabola, or hyperbola). We will be examine only elliptical orbits for this analysis. Looking at equations (2) and (3) the parameters "a" and "e" represent the size and shape of the elliptic orbit. The parameter ν is the current position on the ellipse. To fully define a satellite orbit we can use the "Classical Orbital Elements": the semi-major axis "a", the eccentricity "e", the inclination "i", the right ascension of ascending node (or longitude of ascending node) Ω, argument of perigee ω, and true anomaly ν to define an orbit.[4] For certain orbits Ω, ω, or ν may be undefined and are replaced with different parameters (see *Space Mission Analysis and Design* for a full description of the orbital elements).[5] Figure 1 below shows the relationship between a satellite orbit, the orbital elements, and the Geocentric-equatorial coordinate frame, or **IJK** frame (the figure was modified slightly for appearance):[6]

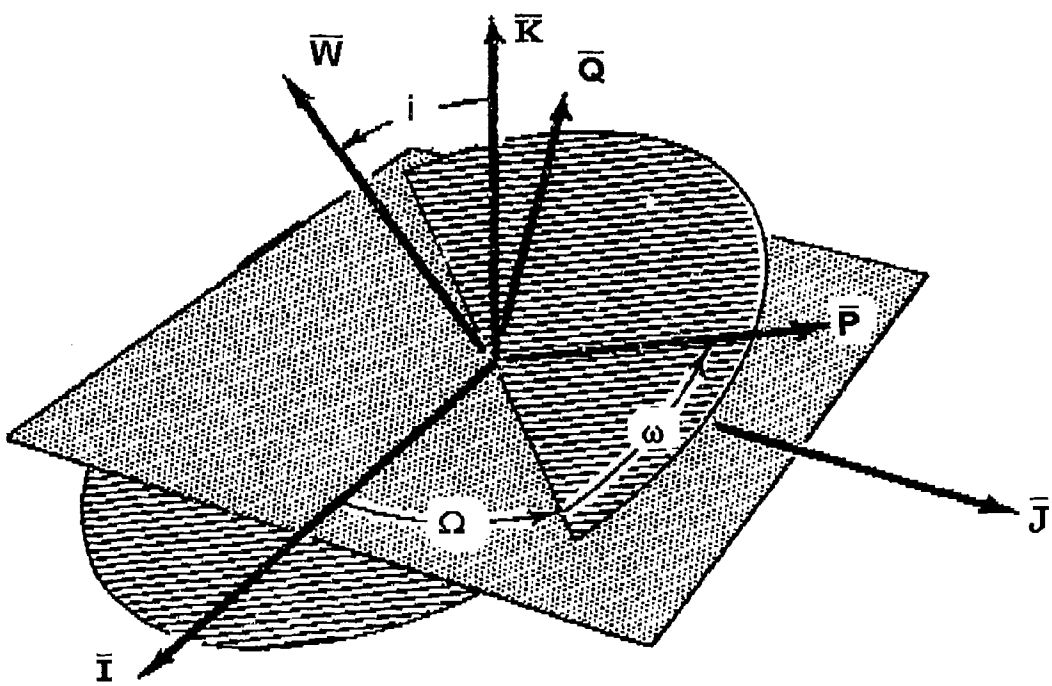
Figure 1 Orbital Elements and **IJK** frame



3.2 *Orbital Elements to Position Vector*

Our initial task is to transform the orbital elements to a position vector in the **IJK** frame. Once we have the satellite's position vector in the **IJK** frame, we should be able to transform this to a latitude and longitude. The first step in this process is to define the orbit in the Perifocal or **PQW** frame. The **PQW** coordinate system as it relates to the **IJK** frame is shown in Figure 2 below:[7]

Figure 2 Relationship between **PQW** and **IJK**



The frame is defined with **P** in the direction of perigee from the center of the Earth, **Q** is in the orbit plane 90 degrees from **P** in the direction of satellite motion, and **W** is perpendicular to the orbit plane to form the third axis of a right hand system.

The following equations relate the orbital elements to the position and velocity vectors in the **PQW** frame.[8]

$$\bar{\mathbf{r}} = r\cos\nu\,\mathbf{P} + r\sin\nu\,\mathbf{Q} \tag{4}$$

$$\bar{\mathbf{v}} = \sqrt{\frac{\mu}{p}}\,[-\sin\nu\,\mathbf{P} + (e + \cos\nu)\mathbf{Q}] \tag{5}$$

Given the orbital elements "a", "e", and ν and using equations (2) through (5) we can find the position and velocity vectors $\bar{\mathbf{r}}$ and $\bar{\mathbf{v}}$ in the **PQW** frame.

Now we need to convert the position and velocity vectors from the **PQW** to the **IJK** frame. Thus we must perform a coordinate system transformation. As Figure 1 shows, the orbital plane is related to the **IJK** frame by the angles "i", Ω, and ω. Using these three angles, we can transform any vector defined in the **PQW** frame to a vector defined in the **IJK** frame (see Appendix A for the coordinate system transformation).[9]

3.3 Orbit Propagation

3.3.1 Kepler Problem

So far we can define a satellite's orbit through the use of the orbital elements and transform a set of elements to a position vector in the **IJK** frame. Now we must find a way to propagate the satellite through its orbit. That is, we must be able to find the satellite's position vector at any time after the initial given position.

The problem of finding a satellite's position in an orbit after some period of time is known as the Kepler problem. There are several sources which go into great detail deriving the solution to the Kepler problem. Here we will go right into the formulas and algorithms for solving this problem. The following is a summary of the information from Fundamentals of Astrodynamics and the reader can refer to chapter 4 of this book for a more detailed analysis.[10]

Out of the six orbital elements, only ν, the true anomaly, changes with time (using the two-body spherical earth assumptions). Thus we need an algorithm which will give us the true anomaly of the satellite after some time period. Thus if we are given a value for ν at T=0, we want to find the new ν at other time.

The following equations are necessary for solving this problem:

$$\cos E = \frac{e + \cos \nu}{1 + e \cos \nu} \tag{6}$$

$$M = E - e \sin E \tag{7}$$

$$n = \sqrt{\frac{\mu}{a^3}} \tag{8}$$

$$M_1 = M + n\Delta T \tag{9}$$

$$\cos \nu_1 = \frac{e - \cos E_1}{e \cos E_1 - 1} \tag{10}$$

The new terms are: eccentric anomaly, E; mean anomaly, M; mean motion, n; and the elapsed time, ΔT. We will assume the initial orbital elements define the orbit and satellite location at a particular time T=0. In equations (6) through (10) the variables M, E, ν are for T=0 and the same variables with a subscript 1 are for a particular time ΔT relative to the initial condition time. Equations (6) through (10) appear to be straight forward. We calculate the mean motion and the eccentric and mean anomalies for T=0 from the initial orbital elements. Then using

equation (9) we get the mean anomaly for the time of interest, M_1. Once we calculate E_1, we can easily find ν_1. After comparing M_1 to ν_1 we can choose the correct ν_1 (remember the inverse cosine function will provide two answers, the true anomaly and eccentric anomaly will both be between 0 and 180 or between 180 and 360 degrees). Thus, we can find ν_1 for any time.

Of course the calculation of the eccentric anomaly from mean anomaly is not as easy as the reverse. Referring to equation (6), we can see there is no way to isolate the eccentric anomaly. This is known as a transcendental function.[11] One way to solve for eccentric anomaly given the mean anomaly is to iterate. That is, pick a value for eccentric anomaly and see what value for mean anomaly you get. We iterate until the difference from the given and calculated mean anomalies is sufficiently small. We can implement the following algorithm easily on a computer:

1 Calculate E, M, n, then M_1 based on ΔT

2. Set $E_1 = M_1$ as an initial guess

3. $E_{NEW} = M_1 + e \, Sin(E_1)$

4. Error = Absolute value of $(E_{NEW} - E_1)$

5. Set $E_1 = E_{NEW}$, this is now the new estimate for E_1.

6. Perform steps 3 through 5 until the error is less than some appropriately small value.

7. Use equation (10) to calculate ν_1. If M_1 is in quadrant 1 or 2 then ν_1 is the angle from the inverse cosine function. If M_1 is in quadrant 3 or 4 then we must take 360 minus the angle we get from the inverse cosine function to get the proper ν_1.

Thus, if we define a set of orbital elements at some epoch time (T=0), we can find ν_1 for any time using the algorithm above. However, some of our two-body assumptions may cause errors in the orbit propagation. To improve our orbit propagator, we will try to account for some of these errors by looking at perturbations.

3.3.2 Perturbations

As we saw in section 3.3, the earth is not a perfect sphere. This not only presents a problem for finding latitude, it impacts the motion of the satellite. For a spherical Earth, the force due to gravity would only depend on the distance of the satellite from the center of the **IJK** frame (which is at the center of the "spherical earth"). Since the earth is oblate, the force due to gravity will vary depending on where the satellite is even if the distance from the origin of the **IJK** frame is the same. Thus the orbit will have a torque resulting from the offset center of gravity.

The result on the two-body equations of motion is that the orbit will precess. The effect on the orbital elements is that Ω and ω will change over time. The following equations describe the change in these elements over time to a degree accurate enough for our purposes:[12]

$$\dot{\Omega}_{J_2} = -1.5\, n\, J_2\, (R_e / a)^2 (\cos i)(1 - e^2)^{-2}$$

$$\cong -2.06474 \times 10^{14} a^{-7/2} (\cos i)(1 - e^2)^{-2} \qquad (11)$$

$$\dot{\varpi}_{J_2} = 0.75\, n\, J_2 (R_e / a)^2 \left(4 - 5\sin^2 i\right)(1 - e^2)^{-2}$$

$$\cong 1.03237 \times 10^{14} a^{-7/2} \left(4 - 5\sin^2 i\right)(1 - e^2)^{-2} \qquad (12)$$

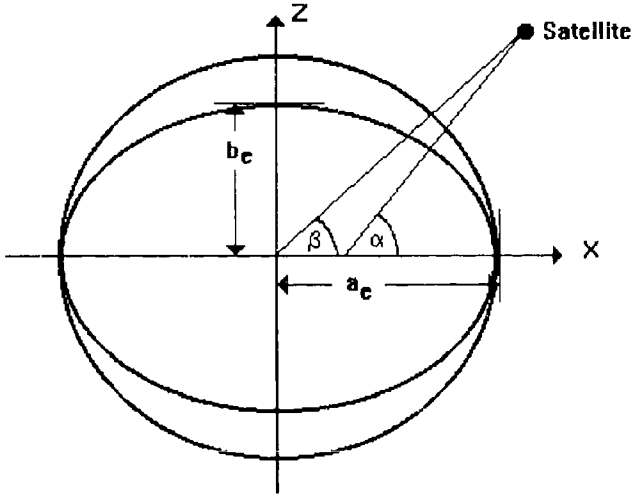
The J_2 term accounts for the earth oblateness and R_e is the Earth's equatorial radius (the same as a_e in section 3.4). There are other J_n terms, known as zonal coefficients, which account for the uneven distribution of mass throughout the earth. However, the J_2 term is responsible for a majority of the perturbations for Earth orbiting satellites.

Thus with the information we have so far, we can easily write code which will propagate a satellite through its orbit. We start with a set of orbital elements for time T=0. For any time of interest we can calculate the new orbital elements Ω, ω, and ν. Next we transform these new elements to a position vector in the **IJK** frame. Now we just need a way to find the subsatellite latitude and longitude. With a series of latitude and longitude coordinates we can form the satellite ground track.

3.4 Position Vector to Geodetic Latitude and Longitude

Now that we can find the position vector from a set of orbital elements at any time, we must find a way to convert the **IJK** position vector to a latitude and longitude on the earth. For a spherical earth, the calculation of a latitude and longitude from the position vector is almost trivial. Unfortunately this is not the case. The earth is oblate. Here we will assume the earth is an ellipse with a polar radius, b_e, of 6356.755 km and an equatorial radius, a_e, of 6378.14 km.[13] The latitude for a spherical earth is known as geocentric while the latitude for an elliptic earth is known as geodetic. Figure 3 below shows the relationship between geodetic and geocentric latitude.[14]

Figure 3 Geodetic versus Geocentric Latitude



α=geodetic latitude β=geocentric latitude

The "X" component of the satellite's position in Figure 3 is formed by taking the resultant of the **I** and **J** components of the position vector. The "Z" component is the same as the **K** component of the position vector. That is:

$$Xo = \sqrt{r_I^2 + r_J^2}\ ,\ Zo = r_K \tag{13}$$

Thus we need to find a way to calculate a latitude and longitude from the components "X" and "Z."

First we must define the angles α and β. The geocentric latitude, β, is the "angle between the equatorial plane and the radius from the geocenter," and geodetic latitude, α, is the "angle between the equatorial plane and the normal to the surface of the ellipsoid." [15] We can see from Figure 3 that the geocentric latitude is simply the inverse tangent of Z/X. However, since most maps use geodetic latitude, it would be useful to find a way to calculate α from the X and Z position vector components.

Suppose we have a point (X, Z) which is on the ellipse and the current position of the satellite is (Xo, Zo). If the slope of the line connecting the two points is also normal to the ellipse at (X, Z), then this line should form the angle α. Pursuing this line of reasoning, the following equations may prove useful:

equation of ellipse $$\frac{X^2}{a_e^2} + \frac{Z^2}{b_e^2} = 1 \qquad (14)$$

slope of line $$\frac{Z - Z_o}{X - X_o} \qquad (15)$$

Slope of normal to point on ellipse $$\frac{a_e^2 Z}{b_e^2 X} \qquad (16)$$

Equation (14) defines the earth ellipsoid. Remembering from basic calculus that the derivative of a function is the tangent or slope function, we can find the slope of the tangent to the ellipse at any point. We can also remember the negative inverse of a slope is the slope of the line perpendicular or normal to the original line. Thus if dZ/dX is the tangent to the ellipse, -dX/dZ is the normal to the ellipse at any (X, Z) which is on the ellipse. By differentiating equation (14) and rearranging terms to get -dX/dZ we arrive at equation (16).

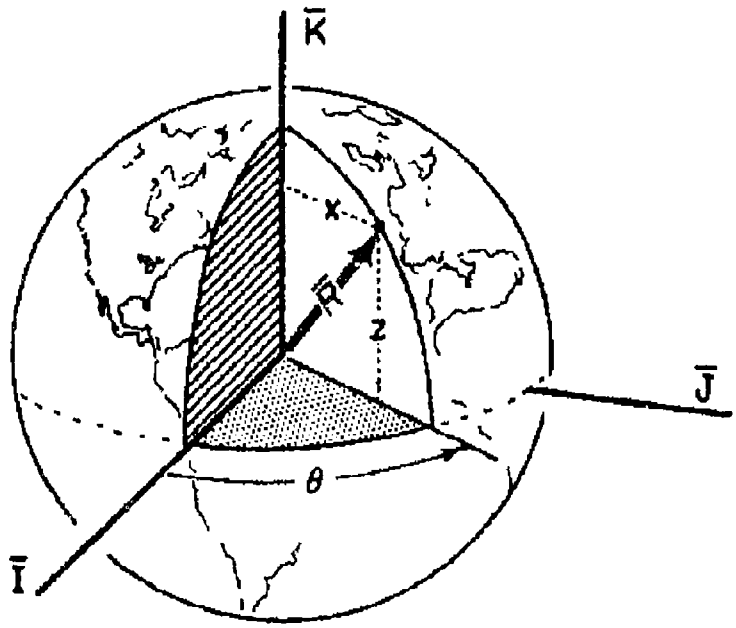
If the location of the satellite is (Xo, Zo), the solution to our problem will be a point (X, Z) which is on the ellipse and allows equations (15) and (16) to be equal. This sounds like an easy problem since this would give us two equations with two unknowns. However, when we solve for one variable and then substitute into equation (14), we end up with a quartic polynomial. See Appendix B for the quartic solution to this problem.

Thus, we need a different approach. Since we are looking for a computer solution, an iteration algorithm seems appropriate. First, we isolate the problem to the first quadrant in the **XZ** frame. This is done by first remembering from equation (13) that Xo must be positive. Since the sign of Zo only determines whether the latitude is north or south, we can limit ourselves to the first quadrant. After we calculate the value of α, we can then go back to Zo to determine if the lati' 'de is north or south.

Now to devise an algorithm to solve for α. Suppose we start off with a first quadrant point (X, Z) which is on the ellipse and a satellite position (Xo, Zo). Plugging these values into equations (15) and (16) will provide the slope of the line from the point to the satellite, or "slope", and the slope of the line normal to the ellipse, or "normal", at (X, Z). If the slope is greater than the normal then our choice for X was too large. If the slope is lower than the normal our choice of X is too low. We can iterate by increasing or decreasing the value of X until the difference between the normal and the slope is very small. The portion of the computer code which accomplishes this algorithm is in Appendix B.

Now with the difficult part complete, we move to the much easier task of finding the longitude. Earlier we found the **X** component of the **XZ** frame position vector from the **I** and **J** components of the **IJK** position vector. If we measure the angle from the **I** axis eastward along the equator to the **X** axis, we get an angle 0 known as "local sidereal time," see Figure 4 below.[16]

Figure 4 Local Sidereal Time and XZ Plane

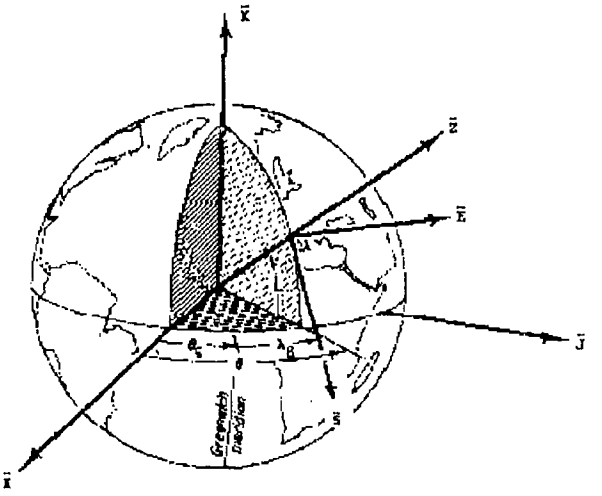


Normally we measure longitude east or west of the Greenwich meridian. Here we will measure the longitude east of Greenwich with the variable λ_E. The angle from the **I** axis to Greenwich is known as "Greenwich sidereal time," θ_g. The following equations and Figure 5 below show how all these variables relate to each other:[17]

$$\theta = \theta_g + \lambda_E \tag{17}$$

$$\theta = \theta_{go} + \varpi_{\oplus}(t - t_o) + \lambda_E \tag{18}$$

Figure 5 GST, LST, and East Longitude Relationship



The parameter $\varpi_{\oplus}$ is the angular velocity of the earth and θ_{go} is the Greenwich sidereal time, or location of Greenwich, at a particular reference time t_o. While we are not using the **SEZ** frame shown in Figure 5, we can view the origin of the **SEZ** frame as the subsatellite point. Additionally the **Z** axis would pass directly through the satellite's position.

From our **IJK** position vector we can easily calculate θ by taking the inverse tangent of r_J/r_I. The sign of the **J** component will give use the proper quadrant. We will make the assumption for software that Greenwich is at the **I** axis at time T=0. In other words, we will initialize the program with θ_{go} at 0. With this assumption, we can solve for λ_E for any time t using equation (18).

This assumption really makes the program more user friendly. We initially select a longitude of ascending node, Ω, by picking a longitude which is appropriate for our target region. After all, our interest is in how the satellite ground track relates to some point on the

ground. Once we pick the initial orbital elements which relate in some way to our target region, the rest will take care of itself. We account for the rotation of the earth after our starting point using the parameter $\varpi_{\oplus}$. Thus we can now calculate the latitude and longitude of the sub-satellite point from the **IJK** position vector.

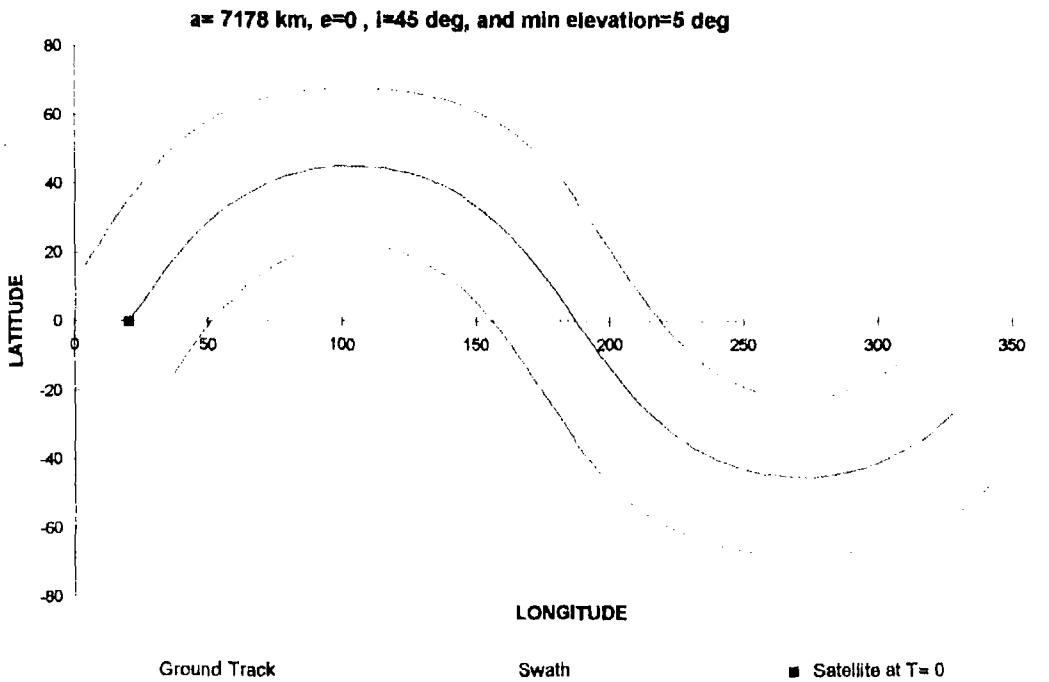
3.5 Satellite Coverage

Generating a ground track from a set of orbital elements may sound complex, but it is not that difficult. A rough sketch of a ground track using only two-body assumptions is a fairly easy task. A more interesting and useful problem is plotting the region which is within the satellite's field of view. The equations developed so far form the necessary basis for allowing us to solve this problem. Here we wish to solve two problems. First, we need to plot the swath ground track. The swath ground track plot shows the area covered by the satellite as it moves through its orbit. Our second problem is to be able to plot the instantaneous field of view (IFOV) at any time. For both of these problems out goal is to show the relationship between the swath ground track and IFOV to a specific ground target region.

3.5.1 Swath Ground Track

What is a swath? The swath is "the area on the surface of the Earth around the ground track that the satellite can observe as it passes overhead."[18] Since a picture is worth a thousand words, we will plot with a ground track and a swath (the region between the outer lines) to help illustrate. In Figure 6 below is a plot of the ground track and swath for the first 90 minutes of a satellite orbit.

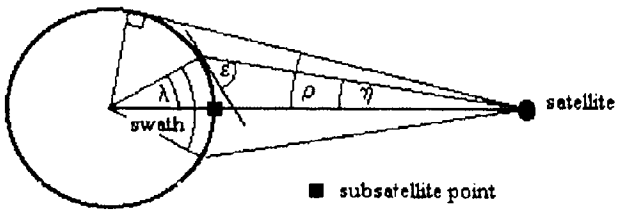
Figure 6 Ground Track and Swath Coverage



We can see from this plot that any latitude and longitude that is within the swath will be visible to the satellite at some time during this pass. Now we must develop the math necessary to generate the swath plot.

First, we must define some angles which relate the satellite's view of the earth to the swath. Below is Figure 7 which shows the geometry and angles we will need for our calculations.[19]

Figure 7 Satellite Viewing Angles



The angle λ is the earth central angle. This happens to be half the swath in terms of earth angle. The angle ε is the elevation angle, or the angle above the horizon the satellite is to an observer on the earth. This will be one of the input parameters for the software. A communications satellite may have an ε of perhaps 5 to 10 degrees while a photo reconnaissance satellite might be limited to 30 to 45 degrees. The nadir angle, or η the angle off the subsatellite point which is to the edge of the field of view. Finally the angle ρ is the angle from the subsatellite point to the horizon.

Initially we will look at the angle λ. From this angle we should be able to find a way to generate the swath. The following formulas will help us find λ from the ε and the position vector of the satellite.[20]

$$\sin\rho = \frac{Re}{Re + H} \tag{19}$$

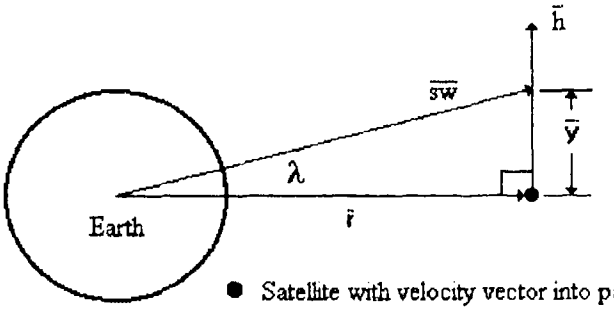
$$\sin\eta = \cos\varepsilon \sin\rho \tag{20}$$

$$\lambda = 90 - \eta - \varepsilon \tag{21}$$

In equation (19), Re is the radius of the Earth and H is the satellite altitude above the Earth. Since Re + H is the same as the magnitude of the position vector, we have all the information necessary to calculate λ.

Now we need a way to find the location of the swath for a given satellite position. Figure 8 below shows the geometry of the earth central angle, λ, with a satellite "flying into the page."

Figure 8 Swath Vector



● Satellite with velocity vector into page

$\overline{sw}$ Swath vector

If we can find the vector $\overline{y}$ in Figure 8, we can simply add $\overline{y}$ to $\overline{r}$ to get the swath vector. If we subtract $\overline{y}$ from $\overline{r}$ we would get the swath vector on the other side as well. Since the velocity vector, $\overline{v}$ is into the page, we can form the vector $\overline{h}$ by taking the cross product of $\overline{r}$ and $\overline{v}$. Then we can use the equations below to find $\overline{y}$.

$$y = |\overline{r}| \tan\lambda \tag{22}$$

$$\overline{y} = y\frac{\overline{h}}{|\overline{h}|} \tag{23}$$

$$\overline{h} = \overline{r} \times \overline{v} \tag{24}$$

Using equations (22) through (24) we can form two swath vectors by adding or subtracting **y** to or from **r**. Since both of these vectors are in the **IJK** frame, we can find the latitude and longitude where these vectors pass through the earth using the algorithms from section 3.4. Plotting a series of such points will provide us a swath ground track.

3.5.2 Instantaneous Field of View

Not only do we want a swath ground track, but we would like to have the instantaneous field of view for any particular time. The swath ground track will allow us to compare a target region on the earth with satellite coverage over time. The instantaneous field of view (IFOV) will allow us to analyze the satellite coverage for any time and compare this with our target region. This will also make it easy to visually see how the coverage varies for a particular pass by stepping through different IFOVs.

This calculation is fairly easy. Since we are dealing with a specific time, we will have a specific satellite position vector. From the methods of section 3.4, we can find the subsatellite latitude and longitude for the time in question. The following equations will allow us to calculate the latitude and longitude (δ_T, L_T) of any point given the azimuth (AZ), earth central angle λ, and the subsatellite latitude and longitude (δ_S, L_S).[21]

$$\Delta L = |L_S - L_T| \tag{25}$$

$$\cos(\delta_T') = \cos\lambda \sin\delta_S + \sin\lambda \cos\delta_S \cos AZ \qquad (\delta_1' < 180) \tag{26}$$

$$\cos\Delta L = (\cos\lambda - \sin\delta_S \sin\delta_T) / (\cos\delta_S \cos\delta_T) \tag{27}$$

$$\delta_T' = 90 - \delta_T \tag{28}$$

Using equations (25) through (28) and varying the azimuth angle from 0 to 360 degrees, we can generated a series of points which are at the edge of the satellite's IFOV.

4. Software

Now that we have all the equations and algorithms from section 3, we have all the tools necessary to write computer code. We want code which will allow us to generate plots of the satellite ground track, swath ground track, IFOV, and even the orbit relative to the earth as viewed from the positive **W** direction of the **PQW** frame. In order to make the software available to many users and possibly students, we are using Visual Basic modules for Microsoft Excel. Additionally, the subroutines can be accessed as user-defined functions. Thus the user can use the algorithms to solve other spreadsheet type problems. Now we will introduce the software.

4.1 Excel Software

We are using Microsoft Excel mainly for the ease of generating user-defined functions. While the user may find this software slow on some machines, the author could not find a more widely available software package to solve this problem. Additionally, the ease of generating plots with Excel and importing these plots to documents made it the software of choice.

Here we will introduce the code for the excel file PLOT.XLS. All the code for PLOT is in Appendix C. Module 1 contains all the number crunching code and algorithms. Module 2 contains the code for the "bells and whistles," or the user friendly dialog boxes. All of the plots are made by calling certain functions from module 1 into several separate spreadsheets. We then form different charts by using the data from the spreadsheets. The modules and spreadsheets for chart generation are all hidden when the user opens the PLOT.XLS file.

PLOT is useful for short time spans and low Earth orbits since it will only generate approximately 75 subsatellite points. For higher orbits, or if the user desires several revolutions for the plots, we are including SLOWPLOT.XLS. SLOWPLOT is the same program, however it will plot 500 points. As the name implies, it is slow. It will take about one minute to generate a plot from a set of orbital elements on a 486-66 machine. PLOT will only require about 10 seconds to generate plots since it is plotting fewer points.

4.2 Software Setup

To get going, simply open the file PLOT.XLS or SLOWPLOT.XLS under Excel. The user will find five chart sheets and one input sheet when first opening the file. Once the user opens the Excel file, it is very important to select the manual calculation mode. From the toolbar select the "Tools" menu bar then select "Options." Click on the "Calculation" tab and select manual calculation. If the user leaves the calculation mode in automatic, Excel will run though all the calculations every time the user changes a number. Since many calculations are necessary, this really slows down the program. There is a button with the label "Too slow?" which will remind the user what steps to take to set the calculation mode to manual. This button is on the "Info" sheet, just click the button and the instructions will appear.

Again, once the file is open, there should be five sheets or charts. Each has a title on the lower tab. The sheets and their contents are:

1. Gnd Track & Inst Cov - This chart shows the target region, the satellite ground track, and the IFOV for any specific time.

2. Inst Cov Vs Target - This chart shows the IFOV for any specific time and the target region.

3. Swath Plot - This chart shows the satellite ground track, the swath track, and the target region.

4. Gnd track - This chart only shows the ground track of the satellite.

5. Orbit - This is a plot showing the orbit relative to the earth. It shows the satellite position at T=0 and the satellite position for any specific time.

6. Info - The is the sheet where the user enters the data. The user may enter the data directly into the spreadsheet or use the "buttons" to the right of the data which will cause the software to prompt the user for the data.

Sheets 1, 2, and 5 have a "View new time instant" button on each chart. If the user clicks this button, the software will prompt the user for a new time instant. This button is very useful for stepping the satellite through a coverage pass of a region. The IFOV for different time instants can provide the user with knowledge of how the coverage changes with time for the pass.

4.3 Input Data

Before the software can generate any plots, the user must enter some information. The minimum information is the orbital elements for T=0, the start and finish times relative to T=0 for the plots, and an instantaneous plot time. The target region information is optional. If the user desires a target region, the user must enter the latitude and longitude for the center of a square target region. Additionally, the user must enter the size of one side of the square region in kilometers.

With respect to the target region, there is one other optional piece of information. The user can select the number of orbits between target region plots. This value is set to 0 if the user does not enter a number. This feature will plot the target region location for two points prior to and after the actual target region location. The spacing between the target region plots is the number of orbits the user enters. We will discuss this more in section 0 when we show some sample plots.

In Figure 9 below is a sample of what the information sheet looks like with actual orbit and target region information.

Figure 9 Input Information Sample

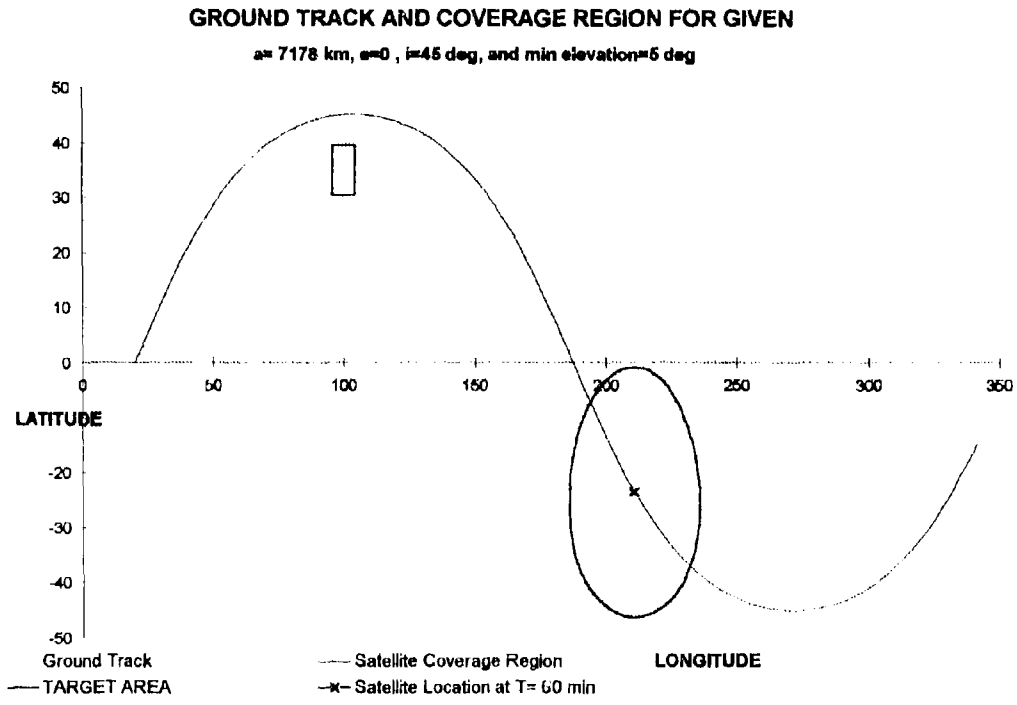
Ground Track Plotter

Information summary		
Initial Orbit Conditions (T=0)		
Semi-major axis	7,178	km
Eccentricity	0	
Inclination	45	deg
Ω (Enter 0 for equatorial orbit)	20	deg
ω , ν, Π (Enter 0 for circular orbit)	0	deg
ν, Uo, or lo at T=0	0	deg
Minimum elevation angle, ε_{min}	5	deg
Start Time for Plot (min before or after T=0)	0	min
Stop Time for Plot (min before or after T=0)	95	min
Time for instantaneous plot	60	min
Target Region Information		
Latitude	35	deg
Longitude	100	deg
Length of side of square grid	1000	km
Number of Orbits between Target region plots	1	
Calculated Period	100.876	min

4.4 Sample Plots

Using the information from Figure 9, we will now show what the some of the plots look like. First, we will start with the chart, the "Gnd Track & Inst Cov" in Figure 10 below.

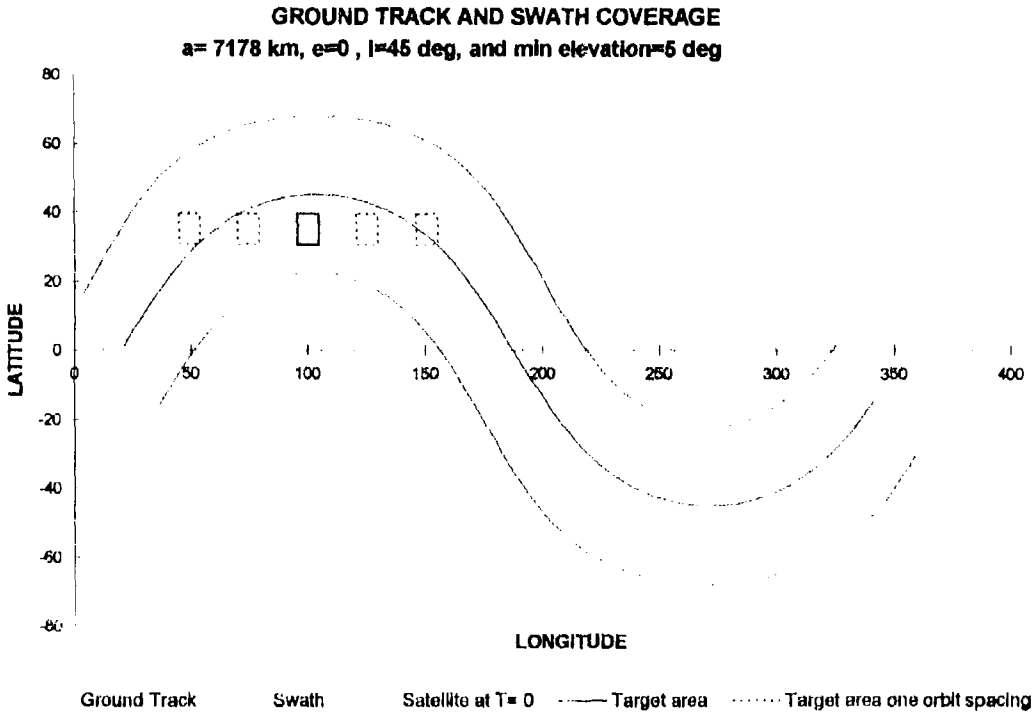
Figure 10 Ground Track & Instantaneous Coverage Plot



We can easily see from this plot that the target region is not visible to the satellite at T=60 minutes. However, the region looks as if it was visible to the satellite earlier.

If we look at the swath plot in Figure 11 below, we can easily see the target region is in view of the satellite during this pass.

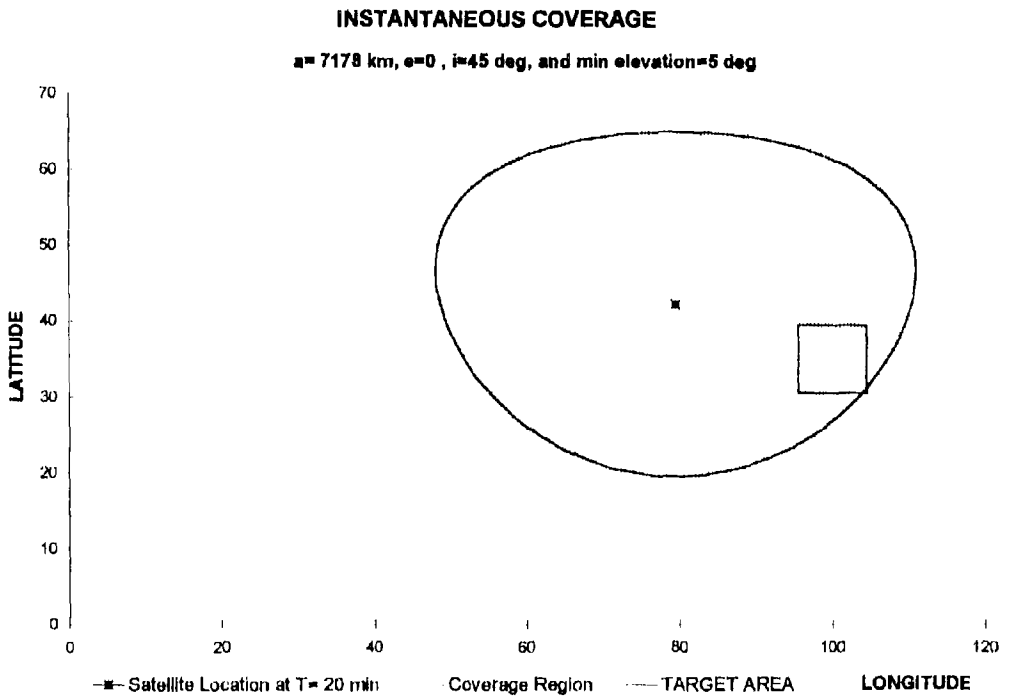
Figure 11 Swath Plot



In fact, we can see the target region was in view for at least the two prior passes and will be in view the next two passes. The target regions with the dashes are where the target region's location is relative to the satellite orbit on other passes. Since we are using one orbit spacing for this example, we can see the location of the target region for two passes prior and after this pass. The spacing takes into account the rotation of the earth and node regression during the orbit. The function "Target" in Appendix C is the subroutine which performs the target region calculations.

One nice feature of the program is the ability to view different time instants easily. We simply click the "View new time instant" button and select the time of interest. We did this for T=20 minutes in the "Inst Cov Vs Target" plot in Figure 12 below.

Figure 12 IFOV Vs Target Region for Specific Time

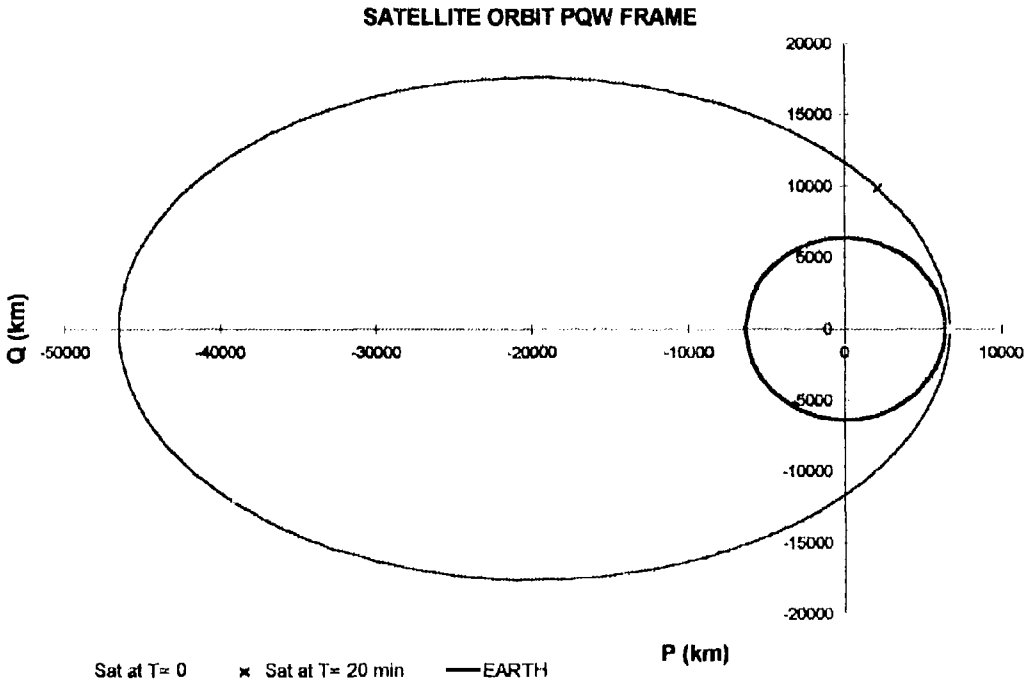


We can easily see from Figure 12 that the entire target region is in view of the satellite at T=20 minutes.

Another plot is the "Gnd Track" plot. We provide ground track plot in case the user only wants the ground track and not the target region, swath, or IFOV information. Finally, the last plot is the orbit relative to the earth. Since this plot shows little useful information for circular

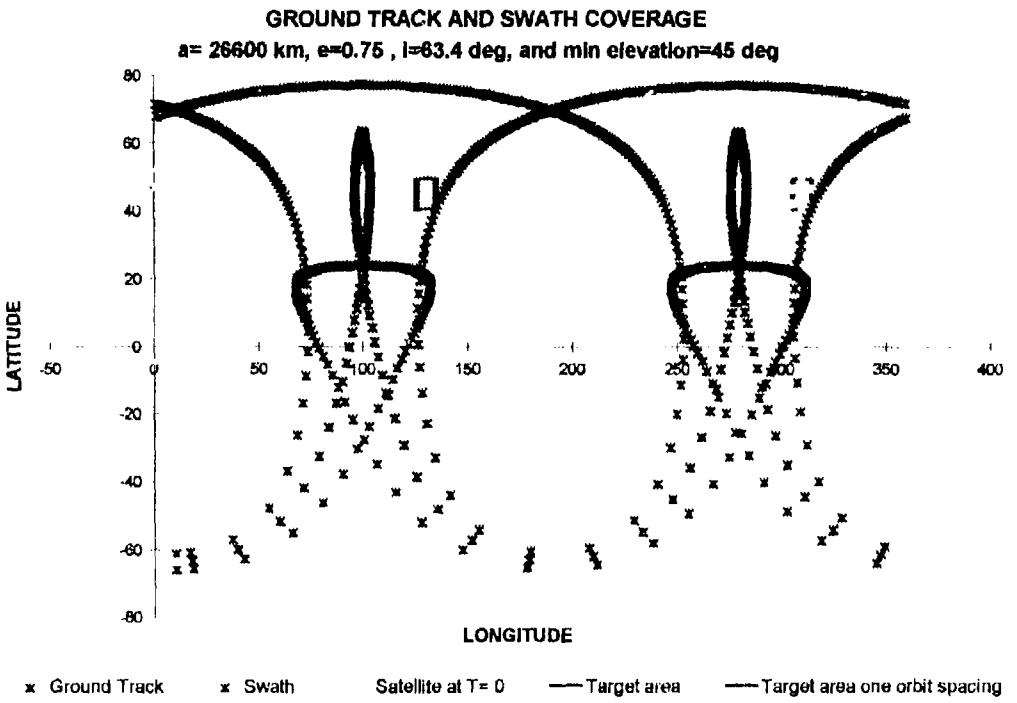
orbits, we will show the plot of a Molniya orbit. In Figure 13 below is the plot for a Molniya orbit with orbital elements of a=26,600, e=0.75, i=63.4 deg, ω=270 deg, Ω=100, and ν=0.[22]

Figure 13 Orbit Relative to Earth



This is one orbit that PLOT does not provide enough points. Since the satellite is flying very fast near perigee, PLOT will miss much of the ground track. Here is where SLOWPLOT can be useful. Since SLOWPLOT uses more points, the ground track will not have the large gap between points that we would have with PLOT. Using SLOWPLOT we can generate the Swath Plot for this same Molniya orbit (see Figure 14 below).

Figure 14 Molniya Plot Using SLOWPLOT.XLS



Now that we now the types of plots that PLOT and SLOWPLOT can generate, we should move on to some limitations.

4.5 Limitations

The main limitation of the software is that it will only work for elliptical orbits. The program will not plot any orbit which is not elliptical (circular orbits are obviously okay since they are just a special case of the ellipse). Additionally, the program is only as good as its assumptions. In summary, we are assuming a two body problem with an oblate Earth. We use

an ellipsoid model for the Earth with a polar radius of 6356.755 km and equatorial radius of 6378.14 km. We are using a value of 398600.5 km^3/sec^2 for the value of μ.[23] Also, the latitude and longitude calculations rely on the accuracy of the spherical earth model. Any differences for a specific location from the ellipsoid model should be considered as necessary.

Since we set our focus on satellite orbits which are "looking" at regions of the Earth, orbits above geosynchronous (GEO) will not have the correct perturbation calculations. Our assumptions only consider the perturbations from the Earth's oblateness. Thus third body effects, such as the earth or moon, which normally become noticeable above GEO, are not considered.

These limitations really have no impact on the graphical data from the various plots. The errors are so small, and the plots are of such a broad scope, the impact is negligible. We mention these for the case where the user decides to use some of the functions for other applications. We will discuss some possible applications in the next section.

4.6 Applications

There are many possible applications where this software can be useful. Many of the functions or subroutines in the software have been used to solve many project related questions. This software is a great as an aid in preliminary design. Different concepts for an orbit can easily be examined using this program. Additionally, this software can be very useful for students learning Astrodynamics. Getting a visual picture of how the orbit, ground track, and satellite coverage changes as we adjust the orbital parameters can be an invaluable tool.

Here is a very interesting application. Suppose you want to run a simulation with several satellites in orbit, and you want to analyze the coverage of a specific region over time. It could be possible to do this all in Excel by modifying the functions and adding a few more. But lets say you want to use a package similar to SIGMA (Simulation Graphical Modeling and Analysis system) to generate the satellite latitudes and longitudes over time. We did this with six lines of code in one node by using a variation of PLOT.XLS. First we eliminate the perturbation effects. Then generate a data file with one period of latitudes and longitudes and the time for each. With this data file you can generate any circular orbit with the same inclination and semi-major axis. It is a simple matter to calculate node regression as a function of time. With a time correction to account for the true anomaly, and a longitude correction to account for node regression, earth rotation, and different Ω's, we can calculate the satellite latitude and longitude for any time.

Thus the tools are here. With a little imagination and some minor changes, they can provide solutions to many problems. Again, the main goal here is to provide some help with the initial designs of an orbit. Use these tools to get you in the ballpark, then use the more powerful programs for the refinement.

5. Conclusion

Many design problems require finding a satellite orbit which can satisfy different constraints. The software packages PLOT.XLS and SLOWPLOT.XLS allow us to visually examine the impact of various orbit parameters on the satellite's coverage. Such a tool can be invaluable when trying to balance out competing parameters. We can easily see how changes to the design will impact coverage of selected regions on the Earth. Finally, the user can access the

various functions in the software for use in other applications. These tools have proven to be very helpful in analyzing various offboard sensor projects in the past. It is up to you to find the uses in the future.

Endnotes

[1] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24.

[2] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p. 130.

[3] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 4, 14.

[4] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p. 135.

[5] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, pp. 133-134.

[6] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24, pp. 55, 59.

[7] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24, pp. 55, 57, 80.

[8] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24, pp. 72-73.

[9] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24, pp. 74-83.

[10] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 212-222.

[11] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p 138.

[12] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p.141.

[13] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p. 807.

[14] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24, pp. 94-95.

[15] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 14, 20, 24, p. 94.

[16] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, p. 99.

[17] Bate, R. R., Mueller, D. D., and White, J. E., *Fundamentals of Astrodynamics*, 2nd ed., Dover Publications, Inc., New York, 1971, pp. 99-100.

[18] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, . p. 113.

[19] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p. 110.

[20] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, pp. 110-111.

[21] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p. 111.

[22] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p. 137.

[23] Larson, W. J. and Wertz, J. R., *Space Mission Analysis and Design*, 2nd ed., Microcosm and Kluwer Academic Publishers, California and Dordrecht, Netherlands, 1993, p.130.

Appendix A Coordinate Transformation PQW to IJK

Here we will show the transformation matrices for converting a vector from the **PQW** frame to the **IJK** frame. As Figure 2 from section 3.2 shows, the **PQW** frame relates to the **IJK** frame through the angle Ω, i, and ω. Looking at Figure 2, we see we must first rotate the **PQW** frame by the angle $-\omega$ about the **W** axis (or 3 axis). Then we rotate the **PQW** frame about the **P** axis (or 1 axis) by the angle -i. Finally we rotate the frame one more time about the **W** axis by an angle $-\Omega$. This will result in the two frames being coincident. These rotation are accomplished using rotation matrices. The following is the mathematical process for transforming a **PQW** vector to an **IJK** vector.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} = \text{rotation about 1 axis by angle } \theta$$

$$\begin{bmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \text{rotation about 3 axis by angle } \beta$$

$$\begin{bmatrix} r_I \\ r_J \\ r_K \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\Omega) & \sin(-\Omega) \\ 0 & -\sin(-\Omega) & \cos(-\Omega) \end{bmatrix} \begin{bmatrix} \cos(-i) & \sin(-i) & 0 \\ -\sin(-i) & \cos(-i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-\varpi) & \sin(-\varpi) \\ 0 & -\sin(-\varpi) & \cos(-\varpi) \end{bmatrix} \begin{bmatrix} r_P \\ r_Q \\ r_W \end{bmatrix}$$

Appendix B Geodetic Latitude Solutions

Quartic Solution

The following is the derivation of the quartic solution for the problem of converting a satellite position to geodetic latitude. First we convert the **IJK** position vector to the **XZ** plane:

$$Xo = \sqrt{r_i^2 + r_j^2} \qquad Zo = r_k$$

Then we find a point (X, Z) on the Earth ellipsoid such that the slope of the line form (Xo, Zo) to (X, Z) is the same as the slope of the line normal to the ellipse at (X, Z).

Equation for the Earth ellipsoid: $$\frac{X^2}{Ae^2} + \frac{Z^2}{Be^2} = 1$$

Equation for the slope of line normal to the ellipse (-dX/dZ) : $$\frac{Ae^2 Z}{Be^2 X}$$

Slope of line from point on ellipse to satellite: $$\frac{Z - Zo}{X - Xo}$$

Set the two slope equations equal to each other and isolate X:

$$\frac{Z - Zo}{Z - Xo} = \frac{Ae^2 Z}{Be^2 X} \qquad X = -\frac{Xo\, Ae^2 Z}{\left(Be^2 - Ae^2\right) Z - ZoBe^2}$$

Now square X and plug into ellipse equation:

$$X^2 = \frac{Xo^2 Ae^4 Z^2}{\left(\left(Be^2 - Ae^2\right) Z - ZoBe^2\right)^2} \qquad \frac{Xo^2 Ae^2 Z^2}{\left(\left(Be^2 - Ae^2\right) Z - ZoBe^2\right)^2} + \frac{Z^2}{Be^2} = 1$$

Solve for Z:

$$\begin{aligned}&\left(Be^4 - 2\,Be^2\,Ae^2 + Ae^4\right)\;Z^4 + \left(2\,Ae^2\,ZoBe^2 - 2\,Be^2\,ZoBe^2\right)\;Z^3\\&+\left(Zo^2 + Xo^2\,Ae^2\,Be^2 - Be^2\,Ae^4 - Be^6 + 2\,Be^4\,Ae^2\right)\;Z^2\\&+\left(2\,Be^4\,ZoBe^2 - 2\,Be^2\,Ae^2\,ZoBe^2\right)\;Z - Be^2\,ZoBe^4 = 0\end{aligned}$$

The correct Z will be a real solution to the quartic which has the same sign as Zo. After we find Z it is a simple matter of substitution to find X. Finally, the geodetic latitude is the inverse tangent of (Z-Zo)/(X-Xo).

Iterative Software solution

The following is the Visual Basic code used in the modules for the Excel program to solve this problem using iteration

```
'TO FIND GEODETIC LATITUDE FROM r IN ijk FRAME
Function LAT(RI, RJ, RK)
  Dim r, X, Z, SLP, NOR, INC As Single
  STP = 0
  CHN = 0
  AE = 6378.14
  BE = 6356.785
  r = mag(RI, RJ, RK)
  INC = 3#
  'CHECK FOR LAT CLOSE TO EQUATOR OR POLE
  'There is little difference between geodetic and geocentric latitude
  'close to these extremes.  Here we will assume they are equal when
  'within 5 deg of the equator or pole.
  If (Abs(ASINR(RK / r) * 180 / PI()) < 5) Or (Abs(ASINR(RK / r) * 180 / PI()) > 85) Then
    LAT = ASINR(RK / r) * 180 / PI()
  Else
    'We will start with an initial guess for X then iterate to solve
    'start with X for geocentric calculation as the initial guess
    X = AE * Sqr(RI ^ 2 + RJ ^ 2) / r
    Z = Sqr(BE ^ 2 * (1 - X ^ 2 / AE ^ 2))
    SLP = (Z - Abs(RK)) / (X - Sqr(RI ^ 2 + RJ ^ 2))
    NOR = (Z * AE ^ 2 / (X * BE ^ 2))
    If SLP > NOR Then CHN = 1 'X is too large so set flag to decrease X
    Do
      If Abs(SLP - NOR) > 0.0001 Then
        'slope too small so increase X
        If ((SLP < NOR) And CHN = 0) Then X = X + INC
```

```
            'slope too small but we decreased X last iteration
            If ((SLP < NOR) And CHN = 1) Then
               CHN = O
               INC = INC / 3#
               X = X + INC
            End If
            'slope too large and we increased X last iteration
            If ((SLP > NOR) And CHN = 0) Then
               CHN = 1
               INC = INC / 3#
               X = X - INC
            End If
            'slope too large so decrease X
            If ((SLP > NOR) And CHN = 1) Then X = X - INC
         Else
            'error is small enough so calculate latitude
            LAT = Atn(NOR) * 180 / PI()
            'check for north(+) or south(-) latitude
            If (RK < 0) Then LAT = -LAT
            STP = 1
         End If
         'X should never be larger than radius of Earth
         If X > AE Then X = AE
         Z = Sqr(BE ^ 2 * (1 - X ^ 2 / AE ^ 2))
         SLP = (Z - Abs(RK)) / (X - Sqr(RI ^ 2 + RJ ^ 2))
         NOR = Z * AE ^ 2 / (X * BE ^ 2)
      Loop While STP = 0
   End If
End Function
```

Comparision of Analytic with Iterative Solution

When we call this function a position vector of r=(7000 I, 7000 J, 7000 K), we get a latitude of 35.36113. This is using an error between the normal and the slope of 0.0001. On the next page we have the analytical solution to the quartic equation using Maple V. Using Maple V we get a latitude of 35.35972729 degrees. This represents a 0.004% error. Since the errors from our other assumptions will probably introduce more error than this, we should be safe at this accuracy.

Maple V Solution For Specific Example

Position vector IJK frame (7000,7000,7000)

```
Ae:=6378.14;Be:=6356.755;Xo:=9899.495;Zo:=7000;
```

$$Ae := 6378.14$$

$$Be := 6356.755$$

$$Xo := 9899.495$$

$$Zo := 7000$$

```
ellipse:=X^2 /Ae^2 + Z^2 / Be^2=1 ;slope_contraint:= (Z-Zo)/(X-Xo)-(Ae^2/
Be^2)*Z/X;
```

$$ellipse := \frac{X^2}{Ae^2} + \frac{Z^2}{Be^2} = 1$$

$$slope\_contraint := \frac{Z - Zo}{X - Xo} - \frac{Ae^2\, Z}{Be^2\, X}$$

```
s:=solve({ellipse,slope_contraint},{X,Z});
```

$$s := \left\{ Z = -.1564955393\ 10^7,\ X = .4396945462\ 10^7 \right\},$$
$$\{ Z = 3670.495837,\ X = 5207.446490 \},$$
$$\{ Z = -3653.243512,\ X = -5219.645701 \}$$

```
lat:=arctan((Zo-3670.495837)/(Xo-5207.446490))*180/3.141592654;
```

$$lat := 35.35972729$$

Appendix C Visual Basic Code

Module 1: Calculation Code For PLOT.XLS

```
'to find the magnitude of a vector
Function mag(a, B, C)
   mag = Sqr(a * a + B * B + C * C)
End Function

'to find the cross product of two vectors
Function CROSS(a, B, C, D, e, F)
   CROSS = Array(B * F - C * e, C * D - a * F, a * e - B * D)
End Function

'to find the dot product of two vectors
Function DOT(a, B, C, D, e, F)
   DOT = a * D + B * e + C * F
End Function

'to find the angle between two vectors
Function ANGLE(a, B, C, D, e, F)
   ANGLE = 180 / PI() * ACOSR(DOT(a, B, C, D, e, F) / (mag(a, B, C) * mag(D, e, F)))
End Function

'to find the inverse cos in radians FOR THIS MODULE ONLY
Function ACOSR(X)
   If (-X * X + 1) <= 0 Then
      ACOSR = 0
   Else
      ACOSR = Atn(-X / Sqr(-X * X + 1)) + PI() / 2
   End If
End Function

'to find the inverse SIN in radians FOR THIS MODULE ONLY
Function ASINR(X)
   If (-X * X + 1) <= 0 Then
      ASINR = PI() / 2
   Else
      ASINR = Atn(X / Sqr(-X * X + 1))
   End If
End Function
```

```
'TO FIND RPQW FROM ORBITAL ELEMENTS
Function RPQW(a, e, truean)
   Dim ANG, r As Single
   ANG = truean * PI() / 180
   r = a * (1 - e ^ 2) / (1 + e * Cos(ANG))
   RPQW = Array(r * Cos(ANG), r * Sin(ANG), 0)
End Function

'TO FIND VPQW FROM ORBITAL ELEMENTS
Function VPQW(a, e, truean)
   Dim ANG, P, C1 As Single
   ANG = truean * PI() / 180
   P = a * (1 - e ^ 2)
   C1 = (398600.5 / P) ^ 0.5
   VPQW = Array(-C1 * Sin(ANG), C1 * (e + Cos(ANG)), 0)
End Function


'PERFORM COORDINATE TRANSFORMATION ANGLES IN DEGREES
'AXIS 1 ROTATION
Function RO  1(a, B, C, ANG)
   Dim X As Single
   X = ANG * PI() / 180#
   ROT1 = Array(a, B * Cos(X) + C * Sin(X), -B * Sin(X) + C * Cos(X))
End Function

'AXIS 2 ROTATION
Function ROT2(a, B, C, ANG)
   Dim Y As Single
   Y = ANG * PI() / 180#
   ROT2 = Array(a * Cos(Y) - C * Sin(Y), B, a * Sin(Y) + C * Cos(Y))
End Function

'AXIS 3 ROTATION
Function ROT3(a, B, C, ANG)
   Dim Z As Single
   Z = ANG * PI() / 180#
   ROT3 = Array(a * Cos(Z) + B * Sin(Z), -a * Sin(Z) + B * Cos(Z), C)
End Function

Function PI()
   PI = 3.14159265358979
End Function
```

```
'PERFORM COORDINATE TRANSFORMATION ANGLES IN DEGREES
' THE FIRST ROTATION IS ABOUT THE 3 AXIS BY A DEGREES
' THE SECOND ROTATION ABOUT THE 1 AXIS BY B DEGREES
' THE THIRD ROTATION ABOUT THE 3 AXIS BY C DEGREES

Function ROT313(RP, RQ, RW, a, B, C)
   temp = ROT3(RP, RQ, RW, a)
   TEMP2 = ROT1(temp(0), temp(1), temp(2), B)
   ROT313 = ROT3(TEMP2(0), TEMP2(1), TEMP2(2), C)
End Function


'TO FIND THE VECTOR DEFINING THE SWATH PERPENDICULAR TO GROUND
TRACK
Function SWATH(RI, RJ, RK, VI, VJ, VK, EL)
   Dim elrad, r, nadir, lambda, MAGY, MAGH, C As Single
   elrad = EL * PI() / 180
   r = mag(RI, RJ, RK)
   nadir = ASINR(Cos(elrad) * 6378.145 / r)
   lambda = PI() / 2 - elrad - nadir
   MAGY = r * Tan(lambda)
   H = CROSS(RI, RJ, RK, VI, VJ, VK)
   MAGH = mag(H(0), H(1), H(2))
   C = MAGY / MAGH
   SWATH = Array(C * H(0), C * H(1), C * H(2))
End Function

'This function will take any angle in degrees and reduce it to an angle
'between 0 and 360 degrees
Function MODULUS(numer, denom)
   Dim REALVALUE As Single
   'perform integer division first
   intvalue = numer \ denom
   REALVALUE = numer / denom
   MODULUS = (Abs(REALVALUE) - Abs(intvalue)) * denom
   If REALVALUE < 0 Then MODULUS = -MODULUS
End Function

'This function will solve the transcendental equation from the Kepler
'problem.  That is, given mean anomaly and eccentricity,
'find the eccentric anomaly
Function ECCNEW(M1, e)
Dim ERR, E1 As Single
```

```
  E1 = M1
  Do
    ECCNEW = M1 + e * Sin(E1)
    ERR = Abs(ECCNEW - E1)
    E1 = ECCNEW
  Loop Until (ERR < 0.0001)
  If M1 > PI() And ECCNEW < PI() Then ECCNEW = 2 * PI() - ECCNEW
End Function

'This will solve the Kepler problem
'THIS WILL RETURN THE VALUE OF THE NEW TRUE ANAMOLY AFTER TIME 'IN
SECONDS
Function KEPLER(a, e, TNOM, time)
  Dim v, ENOM, N, M, MNEW, VNEW, ENEW As Single
  v = TNOM * PI() / 180#
  ENOM = ACOSR((e + Cos(v)) / (1 + e * Cos(v)))
  If v > PI() And ENOM < PI() Then ENOM = 2 * PI() - ENOM
  N = Sqr(398600.5 / a ^ 3)
  M = ENOM - e * Sin(ENOM)
  MNEW = M + N * time    'TIME MUST BE IN SECONDS
  MNEW = MODULUS(MNEW, 2 * PI())
  ENEW = ECCNEW(MNEW, e)
  VNEW = 180 / PI() * ACOSR((e - Cos(ENEW)) / (e * Cos(ENEW) - 1))
  If (ENEW > PI() Or (ENEW > -PI() And ENEW < 0)) And VNEW < 180# Then
    KEPLER = 360# - VNEW
  Else
    KEPLER = VNEW
  End If
End Function

'this will convert orbital elements to position and velocity vector in
'the IJK frame for given time T in minutes
Function rvattime(semi, e, i, OMEGA, ARGPER, truean, T)
Dim TRUENEW, CONST1, WNEW, ASCENNEW As Single
  TRUENEW    KEPLER(semi, e, truean, T * 60)
  PQWR = RPQW(semi, e, TRUENEW)
  PQWV = VPQW(semi, e, TRUENEW)
  CONST1 = (1 - (e) ^ 2) ^ (-2)
  'THIS WILL CORRECT FOR PERTUBATIONS T IS IN MINUTES
  If e = 0 Then
      WNEW = 0
  Else
      WNEW = ARGPER + 71692361111# * T * (semi) ^ (-3.5) * (4 - 5 * (Sin(i * PI()/ 180))^
2) * CONST1
  End If
```

```
   ASCENNEW = OMEGA - 14338472220# * T * (semi) ^ (-3.5) * Cos(i * PI() / 180) *
CONST1
   'Now change to IJK frame,do first rotation with -WNEW, argument of perigee
   'do the second rottion with -inclination
   'do the third rotation with ASCENNEW, the longitude of acsending node
   rijk = ROT313(PQWR(0), PQWR(1), PQWR(2), -WNEW, -i, -ASCENNEW)
   vijk = ROT313(PQWV(0), PQWV(1), PQWV(2), -WNEW, -i, -ASCENNEW)
   rvattime = Array(rijk(0), rijk(1), rijk(2), vijk(0), vijk(1), vijk(2))
End Function

'this will generate the Latitude and Longitude for the subsatellite point
'and the swath for the time interval TI to TF (minutes) for the given elevation
'angle elev.  Time is considered 0 at the given orbital elements.
Function ORBIT(semi, e, i, OMEGA, ARGPER, truean, TI, TF, elev)
Dim TEST(74, 6), TRUENEW, CONST1, WNEW, ASCENNEW, INC, T As Single
INC = (TF - TI) / 73#
T = TI
For N = 0 To 73
   temp = rvattime(semi, e, i, OMEGA, ARGPER, truean, T)
   rijk = Array(temp(0), temp(1), temp(2))
   vijk = Array(temp(3), temp(4), temp(5))
   'Now find the two vectors defining the swath perpendicular to the ground track
   Y = SWATH(rijk(0), rijk(1), rijk(2), vijk(0), vijk(1), vijk(2), elev)
   SW1 = Array(rijk(0) + Y(0), rijk(1) + Y(1), rijk(2) + Y(2))
   SW2 = Array(rijk(0) - Y(0), rijk(1) - Y(1), rijk(2) - Y(2))
   'Now calculate the subsatellite point (SSP) latitude and longitude
   ' for this time T
   'GREENWICH MERIDIAN IS INITIALIZED AT THE I AXIS AT TIME=0
   TEST(N, 0) = T
   TEST(N, 1) = LON(rijk(0), rijk(1), rijk(2), T)
   TEST(N, 2) = LAT(rijk(0), rijk(1), rijk(2))
   'FIND THE LONG AND LAT OF FIRST SWATH VECTOR
   TEST(N, 3) = LON(SW1(0), SW1(1), SW1(2), T)
   TEST(N, 4) = LAT(SW1(0), SW1(1), SW1(2))
   'FIND THE LONG AND LAT OF SECOND SWATH VECTOR
   TEST(N, 5) = LON(SW2(0), SW2(1), SW2(2), T)
   TEST(N, 6) = LAT(SW2(0), SW2(1), SW2(2))
   T = T + INC
Next N
ORBIT = TEST
End Function


'THIS WILL FIND THE INSTANTANEOUS FIELD OF VIEW FOR THE SATELLITE
'AT A PARTICULAR TIME (MINUTES)
```

```
Function instant(semi, e, i, OMEGA, ARGPER, truean, elev, T)
Dim fovloc(73, 2), slat, slon, elrad, r, nadir, lambda As Single
  temp = rvattime(semi, e, i, OMEGA, ARGPER, truean, T)
  rijk = Array(temp(0), temp(1), temp(2))
  slon = LON(rijk(0), rijk(1), rijk(2), T)
  slat = LAT(rijk(0), rijk(1), rijk(2)) * PI() / 180#
  elrad = elev * PI() / 180
  r = mag(rijk(0), rijk(1), rijk(2))
  nadir = ASINR(Cos(elrad) * 6378.145 / r)
  lambda = PI() / 2 - elrad - nadir
  'first find the start with the LON AND LAT of the SSP for this time instant
  fovloc(0, 0) = slon
  fovloc(0, 1) = slat * 180 / PI()
  'CALCULATE THE LON and LAT AROUND THE SSP AT THE DESIRED SWATH
  X = 1
  For AZ = 0 To 360 Step 5
    tlat = PI() / 2 - ACOSR(Cos(lambda) * Sin(slat) + Sin(lambda) * Cos(slat) * Cos(AZ * PI() /
180#))
    DELTALON = ACOSR((Cos(lambda) - Sin(slat) * Sin(tlat)) / (Cos(slat) * Cos(tlat))) * 180 /
PI()
    fovloc(X, 1) = tlat * 180 / PI()
    If AZ <= 180 Then
      fovloc(X, 0) = slon + DELTALON
    Else
      fovloc(X, 0) = slon - DELTALON
    End If
    X = X + 1
  Next AZ
  instant = fovloc
End Function

'THIS WILL GENERATE THE TARGET GRIDS FOR THE DESIRED TIMES
'The input parameter "num" is the number of satellite orbits
'between target grid plots.  Semi is the semi-major axis of the orbit
'which is necessary to determine then period of then satellite
Function target(tlat, tlon, size, semi, num, e, i)
Dim COOR(24, 1), RAD, PER, DEGCH, LUP, LLOW, CENTER As Single
  RAD = (size / 6378.145) * 90 / PI()
  PER = 0.0001658669 * semi ^ (1.5)
  'calculate node regression in degrees per orbit
  nodereg = -23849472.41 * semi ^ (-2) * Cos(i * PI() / 180) * (1 - e ^ 2) ^ (-2)
  'now calculate how far the earth rotates per satellite orbit
  DEGCH = PER * num * 0.2506844773
  LUP = tlat + RAD
  LLOW = tlat - RAD
```

```
  X = 0
  For T = -2 To 2
   'calculate the latitude of the square grid
   COOR(X, 1) = LUP
   COOR(X + 1, 1) = LLOW
   COOR(X + 2, 1) = LLOW
   COOR(X + 3, 1) = LUP
   COOR(X + 4, 1) = LUP
   'calculate the center target longitude is for this satellite pass
   CENTER = tlon + T * (DEGCH + nodereg)
   'get the center longitude between 0 and 360 degrees
   CENTER = MODULUS(CENTER, 360)
   If CENTER < 0 Then CENTER = CENTER + 360
   COOR(X, 0) = CENTER - RAD
   COOR(X + 1, 0) = CENTER - RAD
   COOR(X + 2, 0) = CENTER + RAD
   COOR(X + 3, 0) = CENTER + RAD
   COOR(X + 4, 0) = CENTER - RAD
   X = X + 5
  Next T
  target = COOR
End Function

'this function calculates the earth centrral angle between two
'sets of latitude and longitude
Function MAPANGLE(slat, slon, tlat, tlon)
   SLT = slat * PI() / 180
   Sln = slon * PI() / 180
   TLT = tlat * PI() / 180
   TLN = tlon * PI() / 180
   MAPANGLE = ACOSR(Sin(SLT) * Sin(TLT) + Cos(SLT) * Cos(TLT) * Cos(Abs(Sln -
TLN))) * 180 / PI()
End Function



'TO FIND GEODETIC LATITUDE FROM r IN ijk FRAME
Function LAT(RI, RJ, RK)
   Dim r, X, Z, SLP, NOR, INC As Single
   STP = 0
   CHN = 0
   AE = 6378.14
   BE = 6356.785
   r = mag(RI, RJ, RK)
   INC = 3#
```

```
'CHECK FOR LAT CLOSE TO EQUATOR OR POLE
'There is little difference between geodetic and geocentric latitude
'close to these extremes.  Here we will assume they are equal when
'within 5 deg of the equator or pole.
If (Abs(ASINR(RK / r) * 180 / PI()) < 5) Or (Abs(ASINR(RK / r) * 180 / PI()) > 85) Then
   LAT = ASINR(RK / r) * 180 / PI()
Else
   'We will start with an initial guess for X then iterate to solve
   'start with X for geocentric calculation as the initial guess
   X = AE * Sqr(RI ^ 2 + RJ ^ 2) / r
   Z = Sqr(BE ^ 2 * (1 - X ^ 2 / AE ^ 2))
   SLP = (Z - Abs(RK)) / (X - Sqr(RI ^ 2 + RJ ^ 2))
   NOR = (Z * AE ^ 2 / (X * BE ^ 2))
   If SLP > NOR Then CHN = 1 'X is too large so set flag to decrease X
   Do
      If Abs(SLP - NOR) > 0.0001 Then
         'slope too small so increase X
         If ((SLP < NOR) And CHN = 0) Then X = X + INC
         'slope too small but we decreased X last iteration
         If ((SLP < NOR) And CHN = 1) Then
            CHN = 0
            INC = INC / 3#
            X = X + INC
         End If
         'slope too large and we increased X last iteration
         If ((SLP > NOR) And CHN = 0) Then
            CHN = 1
            INC = INC / 3#
            X = X - INC
         End If
         'slope too large so decrease X
         If ((SLP > NOR) And CHN = 1) Then X = X - INC
      Else
         'error is small enough so calculate latitude
         LAT = Atn(NOR) * 180 / PI()
         'check for north(+) or south(-) latitude
         If (RK < 0) Then LAT = -LAT
         STP = 1
      End If
      'X should never be larger than radius of Earth
      If X > AE Then X = AE
      Z = Sqr(BE ^ 2 * (1 - X ^ 2 / AE ^ 2))
      SLP = (Z - Abs(RK)) / (X - Sqr(RI ^ 2 + RJ ^ 2))
      NOR = Z * AE ^ 2 / (X * BE ^ 2)
   Loop While STP = 0
```

```
  End If
End Function

'TO FIND GEODETIC LONGITUDE FROM r IN IJK FRAME
Function LON(RI, RJ, RK, TIM)
  If RI = 0 Then
    If RJ > 0 Then LON = 90
    If RJ < 0 Then LON = 270
  Else
    LON = Atn(RJ / RI) * 180 / PI()
    If RI < 0 Then LON = LON + 180
    If RI > 0 And LON < 0 Then LON = 360 + LON
  End If
  CORLON = LON - 0.2506844773 * TIM    ' CORRECT FOR ROTATION OF THE EARTH
    If CORLON < 0 Then
      Do
        CORLON = CORLON + 360
      Loop Until CORLON > 0
    End If
  LON = CORLON
End Function
```

Module 2: Dialog Box Code for PLOT.XLS

```
Private a, e, i, om, w, v, TI, TF, tins, LT, LNG, size, numorb, elev
Sub INFO()
  default = 0
  msg1 = "Welcome to the orbit plotter.  "
  msg2 = "Greenwich is initialized at 0 DEG at Time T=0.  "
  msg3 = "Now we will get some information about the satellite orbit.  "
  msg4 = "IS THE ORBIT CIRCULAR?"
  msg = msg1 + msg2 + msg3 + msg4
  Style = vbYesNo
1: response = MsgBox(msg, Style, "ORBIT INFORMATION")
  If response = vbYes Then
    ALT = InputBox("ENTER THE ALTITUDE (KM)", , default)
    a = ALT + 6378.145
    e = 0
    perigee = a * (1 - e)
    If perigee <= 6378.145 Then
      msg1 = "This orbit has a perigee which will impact the earth.  "
      msg2 = "  Retry or quit, this orbit cannot be plotted"
      response = MsgBox(msg1 & msg2, 5, "WARNING!")
      If response = vbRetry Then
        GoTo 1
```

```
    response = MsgBox(msg1 & msg2, 5, "WARNING!")
    If response = vbRetry Then
      GoTo 1
    Else
      GoTo 2
    End If
  End If
  i = InputBox("ENTER THE INCLINATION", , default)
  If i = 0 Then
    om = 0
    w = 0
    v = InputBox("ENTER THE TRUE LONGITUDE AT T=0", , default)
  Else
    om = InputBox("ENTER THE LONGITUDE OF ASCENDING NODE", , default)
    w = 0
    v = InputBox("ENTER THE ARGUMENT OF LATITUDE AT T=0", , default)
  End If
Else
  a = InputBox("ENTER THE SEMI-MAJOR AXIS (KM)", , default)
  e = InputBox("ENTER THE ECCENTRICITY ", , default)
  perigee = a * (1 - e)
  If perigee <= 6378.145 Then
    msg1 = "This orbit has a perigee which will impact the earth. "
    msg2 = "  Retry or quit, this orbit cannot be plotted"
    response = MsgBox(msg1 & msg2, 5, "WARNING!")
    If response = vbRetry Then
      GoTo 1
    Else
      GoTo 2
    End If
  End If
  i = InputBox("ENTER THE INCLINATION", , default)
  If i = 0 Then
    om = 0
    w = InputBox("ENTER THE LONGITUDE OF PERIAPSIS", , default)
    v = InputBox("ENTER THE TRUE ANOMALY AT T=0", , default)
  Else
    om = InputBox("ENTER THE LONGITUDE OF ASCENDING NODE", , default)
    w = InputBox("ENTER THE ARGUMENT OF PERIAPSIS", , default)
    v = InputBox("ENTER THE TRUE ANAMOLY AT T=0", , default)
  End If
End If
period = Int(0.0001658669 * a ^ (1.5))
msg9 = "Your selected orbit has a peri      f  "
msg10 = "Would you like to plot only one orbit beginning at T=0?"
```

```
   msg = msg9 & period & "  minutes.  " & msg10
   response = MsgBox(msg, Style, "PLOT INFORMATION")
   If response = vbYes Then
      TI = 0
      TF = period
   Else
      msg1 = "The times asked for here are the number of minutes "
      msg2 = "before T=0 (negative values) or after T=0 (positive values).  "
      TI = InputBox(msg1 & msg2 & "Enter the start time for the plot.", , default)
      TF = InputBox("Enter the stop time for the plot (in minutes).", , default)
   End If
   msg1 = "One plot shows a snaphot of the satellite field of view for a given time.  "
   tins = InputBox(msg1 & "Enter the time (min) for the snapshot plot.", , default)
   msg1 = "Enter the minimum elevation angle (deg) desired "
   msg2 = " (the minimum angle from the earth horizon to the satellite).  "
   elev = InputBox(msg1 & msg2, , default)
   LT = 0
   LNG = 0
   size = 0
   numorb = 1
   Worksheets("Info Sheet").Range("c5 ") = a
   Worksheets("Info Sheet").Range("c6 ") = e
   Worksheets("Info Sheet").Range("c7 ") = i
   Worksheets("Info Sheet").Range("c8 ") = om
   Worksheets("Info Sheet").Range("c9 ") = w
   Worksheets("Info Sheet").Range("c10 ") = v
   Worksheets("Info Sheet").Range("c11 ") = elev
   Worksheets("Info Sheet").Range("c13 ") = TI
   Worksheets("Info Sheet").Range("c14 ") = TF
   Worksheets("Info Sheet").Range("c15 ") = tins
   Worksheets("Info Sheet").Range("c18 ") = LT
   Worksheets("Info Sheet").Range("c19 ") = LNG
   Worksheets("Info Sheet").Range("c20 ") = size
   Worksheets("Info Sheet").Range("c21 ") = numorb
   msg1 = "Orbit info loaded. You can now enter target info if desired "
   msg2 = "or click on PLOT NOW to see plots.  Click OK to continue."
   r = MsgBox(msg1 & msg2, 0)
2:
End Sub


Sub tgtINFO()
   Style = vbYesNo
   msg1 = "We can put a square grid on the plots representing "
   msg2 = "a region of interest.  Simply enter the latitude and longitude "
```

```
    msg3 = "for the center of the region and the desired length of "
    msg4 = "the grid in kilometers.  Click YES to continue or NO to skip this feature."
    response = MsgBox(msg1 & msg2 & msg3 & msg4, Style, "TARGET INFORMATION")
    If response = vbYes Then
       LT = InputBox("Enter the region Latitude (north is +).")
       LNG = InputBox("Enter the region Longitude measured East of Greenwich.")
       size = InputBox("Enter the length of the square region (km).")
       msg1 = "We can also plot the location of the target region relative to "
       msg2 = "the ground track for previous and subsequent orbits.  "
       msg3 = " Click Yes to use and No to skip this feature."
       response = MsgBox(msg1 & msg2 & msg3, Style, "TARGET INFORMATION")
       If response = vbYes Then
          numorb = InputBox("Enter the number of orbits between relative target region plots")
       Else
          numorb = 0
       End If
       Worksheets("Info Sheet").Range("c18 ") = LT
       Worksheets("Info Sheet").Range("c19 ") = LNG
       Worksheets("Info Sheet").Range("c20 ") = size
       Worksheets("Info Sheet").Range("c21 ") = numorb
       msg1 = "Target info loaded. "
       msg2 = "  When ready to generate plots click on Plot Now. "
       r = MsgBox(msg1 & msg2, 0)
       End If
End Sub

Sub INST()
   tins = Worksheets("Info Sheet").Range("c15 ")
   Worksheets("Info Sheet").Range("c15 ") = InputBox("", "Time for the snapshot plot (min) ",
tins)
   Worksheets("Info Sheet").Calculate
   Worksheets("SHEET1 (2)").Calculate
   Worksheets("SHEET1 (5)").Calculate
End Sub

Sub PLOTNOW()
   a = Worksheets("Info Sheet").Range("c5 ")
   e = Worksheets("Info Sheet").Range("c6 ")
   Debug.Print e
   If a <= 0 Or e < 0 Or e >= 1 Then
      msg1 = "This orbit is not elliptical. "
      msg2 = " Sorry, this program can only plot elliptic orbits."
      response = MsgBox(msg1 & msg2, 0, "WARNING!")
      GoTo 3
   End If
```

```
   perigee = a * (1 - e)
   If perigee <= 6378.145 Then
      msg1 = "This orbit h    perigee which will impact the earth. "
      msg2 = " Check the orbital elements and make the necessary corrections."
      response = MsgBox(msg1 & msg2, 0, "WARNING!")
      GoTo 3
   End If
   Application.Calculate
   Sheets("Swath Plot").Select
3:
End Sub

Sub welcome()
   msg1 = "This program will take orbital element information"
   msg2 = " and make several plots.  The satellite ground track, swath, "
   msg3 = "instantaneous coverage and position in orbit around the Earth "
   msg4 = "for any entered time can be displayed."
   r = MsgBox(msg1 & msg2 & msg3 & msg4, 0)
   msg1 = "Additionally we can plot a target region.  We can then easily"
   msg2 = " compare the ground track and satellite position at a given time"
   msg3 = " to the target region."
   r = MsgBox(msg1 & msg2 & msg3, 0)
   msg1 = "For new users the best way to begin is to click on the ORBIT"
   msg2 = " INTERVIEW button and answer all the questions.  Then click on "
   msg3 = " the ADD TARGET REGION button if you want to enter a target region.  "
   msg4 = "Finally click on PLOT NOW to generate the plots."
   r = MsgBox(msg1 & msg2 & msg3 & msg4, 0)
   msg1 = "The information summary table contains all the parameters you enter."
   msg2 = " You can change any value directly in the table then click the"
   msg3 = " PLOT NOW button to generate the new plots.  "
   r = MsgBox(msg1 & msg2 & msg3, 0)
   msg1 = "Additionally some charts have a VIEW NEW TIME INSTANT button."
   msg2 = " Clicking this button moves the satellite and satellite view "
   msg3 = " to the new time.  This is very useful in comparing the satellite "
   msg4 = " location to the target region at different time instants."
   r = MsgBox(msg1 & msg2 & msg3 & msg4, 0)
   msg1 = "Depending on the speed of your machine, it may take a minute"
   msg2 = " or so to generate the plots.  Be patient, a chart will appear "
   msg3 = "after all calculations are complete.  There will be five different "
   msg4 = "charts, just click on the chart you wish to see."
   r = MsgBox(msg1 & msg2 & msg3 & msg4, 0)
End Sub

Sub slow()
   msg1 = "If you are experiencing long delays after entering data"
```

```
    msg2 = " (except for the PLOT NOW button) there could be "
    msg3 = " an easy fix.  From the TOOLS menu select OPTIONS.  Select"
    msg4 = " Calculation.  Make sure the calculation mode is set to Manual."
    r = MsgBox(msg1 & msg2 & msg3 & msg4 & msg5, 0)
    msg1 = "If this doesn't help, you just have a slow processor."
    msg2 = "  There are numerous calculations being performed, be patient."
    r = MsgBox(msg1 & msg2, 0)
End Sub
```